Layering-ADOPT: ADOPT with Layering Boundary

Chen He*, Ziyu Chen & Zhen He
*Corresponding author: hechen316@163.com
College of Computer Science, Chongqing University, Chongqing, China

ABSTRACT: ADOPT and BnB-ADOPT are the two important search-based complete algorithms to solve Distributed Constraint Optimization Problem (DCOP). However, a large number of solution reconstructions appear in ADOPT and sub-optimal branch can’t be promptly pruned in BnB-ADOPT. A layering DCOP algorithm to combine ADOPT and BnB-ADOPT is proposed to remedy their weakness, named Layering-ADOPT. In Layering-ADOPT, a layering boundary is introduced to divide all agents into two classes, one performing ADOPT and another for BnB-ADOPT. This paper presents a rule to get a layering boundary and a new strategy to realize the joint between ADOPT and BnB-ADOPT. In the experiment, Layering-ADOPT and ADOPT as well as BnB-ADOPT are compared on random DCOPs. The results show that Layering-ADOPT is superior to ADOPT and BnB-ADOPT on the test problems.

Keywords: multi-agent systems; distributed constraint optimization problem; Layering-ADOPT

1 INTRODUCTION

As a promising framework for MAS problems, Distributed Constraint Optimization Problem (DCOP) has become a research hotspot in artificial intelligence area and widely deployed in real life such as distributed task planning [1], meeting scheduling [2] resource allocation [3], sensor network [4] and so on. A DCOP is composed of a number of agents, each responsible for choosing a value from its finite domain for its own variable which is subject to some constraints. Solving a DCOP is to coordinate all the agents and get the best solution to all the constraints. Nowadays, researchers have proposed several distributed search algorithms to solve DCOPs optimally. ADOPT [5] is a representative of all search algorithms with DFS pseudo-tree. It performs search using Best-First Search (BFS) strategy which makes it hopeful for ADOPT to promptly choose an assignment closed to the optimal solution, but may result in unnecessary reconstruction of abandoned solutions. BnB-ADOPT [6] is another important search algorithms. It shares most of the data structures and messages of ADOPT but employs depth-first search (DFS) and branch-and-bound strategy. BnB-ADOPT avoids the solution reconstruction but cannot prune the sub-optimal space efficiently because DFS strategy might not provide good and prompt feedback. Many search algorithms based on ADOPT and BnB-ADOPT have been proposed, like ADOPT* [7], BnB-ADOPT* [8], and BnB-ADOPT*-AC [9]. However, solution reconstruction for ADOPT and prompt feedback for BnB-ADOPT aren’t still solved very well. In order to obtain a better tradeoff between ADOPT and BnB-ADOPT, ADOPT(k) [10] was proposed, which generalized ADOPT and BnB-ADOPT by introducing the parameter k to adjust the value variation conditions.

As can be seen, BFS strategy used in ADOPT can get a better and quick lower-bound feedback at the cost of unnecessary reconstruction of abandoned solutions, meanwhile DFS strategy used in BnB-ADOPT avoids solution reconstruction but can’t promptly prune sub-optimal space. Also, we find that BFS strategy and DFS strategy have different effects corresponding to agents having different positions in the pseudo-tree. If the position of each agent in the pseudo-tree is considered, the two strategies can be compensated for their weaknesses. Therefore, we propose a layering DCOP algorithm by adopting a layering boundary that divides all agents into BFS-based agents and DFS-based agents, named Layering-ADOPT. Different from ADOPT(k), Layering-ADOPT aims to exert prompt response in BFS strategy and no solution
reconstruction in DFS strategy. In Layering-ADOPT, each agent can select its strategy according to its position in the pseudo-tree.

The remainder of this paper is outlined as follows. Section 2 provides the preliminaries. In Section 3, the details of our proposed algorithm, Layering-ADOPT, are illustrated. Section 4 presents the experimental evaluation and discussion of the results. Conclusions and future works are summarized in Section 5.

2 PRELIMINARIES

A DCOP can be represented by a tuple \(<A, X, D, F>\) such that: \(A=\{a_1, a_2, ..., a_n\}\) is a set of agents; \(X=\{x_1, x_2, ..., x_n\}\) is a set of variables, where each variable is assigned to an agent; \(D=\{D_1, D_2, ..., D_n\}\) is a set of finite and discrete domains, where \(D_i\) is the domain of variable \(x_i\); \(F=\{f_1, f_2, ..., f_m\}\) is a set of constraints, where each constraint \(f_i : D_1 \times D_2 \times ... \times D_n \rightarrow \mathbb{R}^+\) specifies a non-negative cost for every possible value combination of a set of variables.

Given this, the goal for the agents is to find the joint variable assignment \(X^*\) such that a given global objective function is minimized. Generally, the objective function is described as the sum over a set of \(f_i\).

To facilitate understanding, this paper assumes that each agent has a single variable and constraints are binary relations. Here, the term “agent” and “variable” can be used interchangeably. A binary constraint is a constraint involving exactly two variables with defined as \(f_i : D_1 \times D_2 \rightarrow \mathbb{R}^+\). The joint variable assignment \(X^*\) is obtained as follows:

\[
X^* = \arg\min_{d_i \in D_i, d_j \in D_j} \sum_{f \in F} f(d_i, d_j)
\]

It is common that a DCOP problem is visualized as a constraint graph where the nodes are the agents and the edges are the constraints. Figure 1 visualizes an example of DCOP problem including constraint graph and constraint costs. In this instance, the minimal sum of all costs is 0 and \(X^*\)={\(x_1=1\), \(x_2=1\), \(x_3=1\), \(x_4=1\), \(x_5=1\), \(x_6=1\), \(x_7=1\)}.

![Figure 1. The example of DCOP.](image)

In order to solve DCOP, most complete DCOP algorithms operate on communication structures based on constraint graph, such as chain and tree structure.

Pseudo-tree [11] is the most commonly used tree structures. Figure 2 shows an example of pseudo-tree deriving from Fig.1. It consists of tree edges shown as solid lines and back edges shown as dashed lines that are not part of the spanning tree (e.g., \(x_3-x_2, x_4-x_7\)).

![Figure 2. A pseudo tree of the constraint graph.](image)
of an agent in the tree (e.g., Depth_{tree}=3 in Figure 2).

Method-the chosen search strategy by each agent.

(2) The rule of layering boundary
A layering boundary is located upon those agents whose depth equal to Depth_{layer} given by the rule of layering boundary. To get a layering boundary Depth_{layer}, we introduce a parameter \( \alpha \in [0,1] \) and Depth_{layer} = \( \lceil \alpha \cdot \text{Depth}_{tree} \rceil \). \( \alpha \) is the ratio of the layer number in pseudo-tree for the two strategies and is related to the problem scale and pseudotree structure. When \( \alpha \) is given, the agents whose depth are less than Depth_{layer} use DFS strategy and others use BFS strategy. Figure 3 and Figure 4 show a constraint graph and its pseudo-tree. In Figure 4, if \( \alpha = 0.4 \), Depth_{layer} = \( 0.4 \times 3 \) = 1. Agent \( x_5 \) chooses DFS strategy with Depth(\( x_5 \)) = 0 < 1 and other agent \( x_i \) chooses BFS strategy with Depth(\( x_i \)) \geq 1.

(3) The joint between DFS and BFS strategy
To realize the joint between DFS and BFS strategy, Layering-ADOPT uses identical data structures and messages as ADOPT. And all agents are divided into two classes: the critical agents which neighbor the layering boundary (Depth(\( x_i \)) = Depth_{layer} or Depth(\( x_i \)) = Depth_{layer} - 1) and the non-critical agents (Depth(\( x_i \)) \neq Depth_{layer} \ W Depth_{layer} - 1). In Figure 4, agent \( x_1, x_3 \) and \( x_5 \) are the critical agents and others are the non-critical agents. The noncritical agents perform DFS or BFS strategy according to their Depth, just like the agents in BnB-ADOPT or ADOPT. The critical agents include the DFS-based bottom-layer agents (Depth(\( x_i \)) = Depth_{layer} - 1) and the BFS based top-layer agents(Depth(\( x_i \)) = Depth_{layer}). The DFS-based bottom-layer agents receive COST messages from their BFS-based children. Because the disposal of COST message is similar in BFS and DFS strategy, the BFS-based bottom-layer agents perform DFS strategy without any change. As is known, the difference between BFS and DFS strategy is reflected in TH_i which is updated by the allocated threshold in VALUE message from parent. And the allocated threshold is an upper bound found for the BFS-based top-layer agents so they can be used to prune. Therefore, a new TH_i^* for the BFS-based top-layer agent is introduced to store the allocated threshold in VALUE message from its parent. The BFS-based top-layer agents will change their values when \( LB_i(d) > \min \{ TH_i, TH_i^* \} \). Besides, the termination condition is TH_i^* = UB_i for the BFS-based top-layer agents in Layering-ADOPT. Because the subtrees rooted at these agents will independently use BFS strategy after receiving the TERMINATE message from their parents and the TH_i^* has become the final optimal cost for the subtrees so far.

3.2 The pseudocode of Layering-ADOPT
The pseudo code is shown in Algorithm 1.

```
procedure Start(arg \( \alpha \))
[01] ChooseMethod();
[02] Contexti := (\( x_{ip} \), ValInit(\( x_{ip} \)), 0) \( x_{ip} \in SCP \};
[03] ID_i := 0;
[04] forall \( x_i \in C(x_i) \ and \ d \in D_i \)
[05] InitChild(\( x_i, d \));
[06] InitSelf();
[07] Backtrack();
[08] if (message queue is not empty)
[09] while (message queue is not empty)
[10] pop msg off message queue;
[11] when Received(msg);
[12] Backtrack();
[13] \( TH_i^* (d) \) := \( h_i^*(d) \);
[14] \( ub_i^* (d) \) := \( \infty \);
[15] \( lb_i^* (d) \) := \( \min \{ TH_i, TH_i^* \} \);

procedure InitSelf()
[16] \( d_i := \arg \min_{d \geq 0} \{ \delta(d) + \sum_{x_{ip} \in SC(x_i)} lb_i^*(d) \} \);
[17] ID_i := ID_i + 1;
[18] if (\( x_i \) is the top layer using BFS method)
[19] \( TH_i^* := \infty \);
[20] \( TH_i := \min_{d \geq 0} \{ \delta(d) + \sum_{x_{ip} \in SC(x_i)} lb_i^*(d) \} \);
```

Figure 3. A constraint graph of a DCOP.

Figure 4. The result of layering boundary in pseudo-tree.
procedure ChooseMethod()
[21] if (Depth(x)) = Depth(root)
[22] Method = DFS;
[23] else
    procedure Backtrack()
[25]forall d ∈ D
[26] LB(d) := δ(d) + \sum \alpha_{x \in Context_i} lb_i^x (d);
[27] UB(d) := \delta(d) + \sum \alpha_{x \in Context_i} ub_i^x (d);
[28] LB := \min_{d \in D} \{ LB(d) \};
[29] UB := \min_{d \in D} \{ UB(d) \};
[31] if (xi uses the BFS method)
[32] else
[33] d := \arg \min_{d \in D} \{ UB(d) \};
[34] else if (xi is the top layer using BFS strategy and
[35] LB(d) > \min\{ TH_i, TH^B_i \})
[36] d := \arg \min_{d \in D} \{ LB(d) \};
[37] else if (LB(d) > TH)
[38] d := \arg \min_{d \in D} \{ LB(d) \};
[39] if (ready to terminate and TH_i = UB_i)
[40] d := \arg \min_{d \in D} \{ LB(d) \};
[41] else if (LB(d) > TH)
[42] d := \arg \min_{d \in D} \{ LB(d) \};
[43] if (a new d has been chosen)
[44] ID_i := ID_i + 1;
[45] MaintainChildThresholdInvariant();
[46] MaintainAllocationInvariant();
[47] if (xi uses the DFS method)
[48] if (TH_i ≤ LB and (xi is a root or ready to terminate))
[49] send(TERMINATE) to each x_i ∈ C(x_i);
[50] terminate execution;
[51] else
[52] if (terminate message received)
[53] if (TH_i = UB_i or (xi is the top layer using BFS
[54] and TH^B_i = UB_i))
[55] send(TERMINATE) to each x_i ∈ C(x_i);
[56] terminate execution;
[57] send(VALUE, x, d, ID_i, \delta(d)) to each x_i ∈ C(x_i);
[58] send(VALUE, x, d, ID_i, \delta(d)) to each x_i ∈ CD(x_i) \setminus C(x_i);
[59] send(COST, x, Context_i, LB_i, UB_i) to pa(x) if xi is not root;
    procedure WhenReceived(TERMINATE)
[59] record terminate message received;
[60] record ready to terminate;
    procedure WhenReceived(VALUE, x, d, ID_i, TH_i)
[61] Context' := Context;
[62] PriorityMerge(x, d, ID_i, Context);
[63] if (!Compatible(Context', Context))
[64] forall x_i ∈ C(x_i) and d ∈ D_i
[65] if (x_p ∈ SCP)
[66] InitChild(x_p, d);
[67] InitSelf();
[68] if (x_p is parent)
[69] if (xi is the top layer using BFS method)
[70] TH_i := TH^B_i;
[71] else
[72] TH_i := TH^B_i;
    procedure WhenReceived(COST, x_p, Context_i, LB_i, UB_i)
[73] Context' := Context;
[74] PriorityMerge(Context_i, Context);
[75] if (!Compatible(Context', Context'))
[76] forall x_c ∈ C(x_c) and d ∈ D_c
[77] if (!Compatible(Context', Context'))
[78] InitChild(x_p, d);
[79] if (Compatible(Context_i, Context))
[80] lb_i^x (d) := \max\{ lb_i^x (d), LB_i \}
[81] for the unique
[82] d := \min\{ lb_i^x (d), UB_i \}
[83] for the unique
[84] if (!Compatible(Context', Context'))
[85] InitSelf();
    procedure MaintainChildThresholdInvariant()
[86] forall x_c ∈ C(x_c) and d ∈ D_c
[87] while (lb_i^x (d) < lb_i^x (d))
[88] wb_i^x (d) := wb_i^x (d) + \epsilon;
[89] while (lb_i^x (d) > ub_i^x (d))
[90] wb_i^x (d) := wb_i^x (d) - \epsilon;
    procedure MaintainAllocationInvariant()
[91] if (xi uses the BFS method)
[92] if (ready to terminate and TH_i = UB_i)
[93] else
[94] th_i^x (d) := TH_i - \delta(d) - \sum \alpha_{x \in Context_i} ub_i^x (d);
[95] else
[96] if (xi is the top layer using BFS strategy)
[97] while \left( \min\{ TH_i^B, TH_i \} > \delta(d) + \sum \alpha_{x \in Context_i} th_i^x (d) \right)
[98] \delta(d) + \sum \alpha_{x \in Context_i} th_i^x (d)
[99] while (TH_i < \delta(d) + \sum \alpha_{x \in Context_i} th_i^x (d))
[100] \delta(d) + \sum \alpha_{x \in Context_i} th_i^x (d)
[101] else
[102] while (TH_i < \delta(d) + \sum \alpha_{x \in Context_i} th_i^x (d))
[103] ;
\[ t^* (d_i) = t^* (d_j) + \epsilon \quad \text{for any } x_i, x_j \in C(x_i) \]
\[ t^* (d_i) > t^* (d_j) \]
\[ t^* (d_i) = t^* (d_j) - \epsilon \quad \text{for any } x_i, x_j \in C(x_i) \]
\[ \text{while } (t^*_i > \delta (d_i) + \sum_{x_i \in x_i} t^*_i (d_i)) \]
\[ \text{for any } x_i \in C(x_i) \]
\[ t^*_i (d_i) = t^*_i (d_j) \]
\[ \text{with } lb^*_i (d_i) < t^*_i (d_i) \]
\[ \text{procedure MaintainThresholdInvarint() \}
\[ \text{if } (TH_i < LB_i) \]
\[ TH_i = LB_i \]
\[ \text{if } (TH_i > UB_i) \]
\[ TH_i = UB_i \]
\[ \text{if } (xi \text{ is the top layer using BFS method}) \]
\[ \text{if } (TH_i > UB_i) \]
\[ TH_i = UB_i \]

Algorithm 1. The pseudocode of Layering-ADOPT

4 EVALUATION

In this section, we compare Layering-ADOPT with ADOPT and BnB-ADOPT. In our experiments, the parameter \( \alpha = 0.5 \) for Layering-ADOPT. Random DCOPs are adopted as the test problems. Random DCOPs include a set of agents randomly constrained with one another. This experimental setup includes unstructured problems where constraints costs are uniformly sampled in the range of 0 to 50. The number of agents, graph density, agents’ domain sizes and cost ranges are varied to assess the algorithms robustness.

We average the experimental results over 50 DCOP problem instances with randomly generated constraints.

We adopt the number of messages (msgNumber), the number of non-concurrent constraint checks (NCCCs) and the real process time (Runtime) as evaluation metrics [6].

Table 1 shows the experimental results that \( nAgents \) varies from 6 to 14 with Domains=3 and density=0.4. Table 2 shows the experimental results that \( nAgents \) varies from 6 to 14 with Domains=3 and density=0.6. Table 3 shows the experimental results that density varies from 0.2 to 0.7 with Domains=3 and \( nAgents=10 \). Table 4 shows the experimental results

<table>
<thead>
<tr>
<th>nAgents</th>
<th>algorithm</th>
<th>communication time=0</th>
<th>communication time=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NCCCs</td>
<td>msgNumber</td>
<td>Runtime</td>
</tr>
<tr>
<td>6</td>
<td>ADOPT</td>
<td>508</td>
<td>606</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>277</td>
<td>273</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>277</td>
<td>273</td>
</tr>
<tr>
<td>8</td>
<td>ADOPT</td>
<td>25744</td>
<td>27355</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>96</td>
<td>1635</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>83</td>
<td>1263</td>
</tr>
<tr>
<td>10</td>
<td>ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>819</td>
<td>21138</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>559</td>
<td>20277</td>
</tr>
<tr>
<td>12</td>
<td>ADOPT</td>
<td>1229</td>
<td>38051</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>913</td>
<td>35175</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>ADOPT</td>
<td>5651</td>
<td>241957</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>4028</td>
<td>230154</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2. The result of varying nAgents with Domains=3 and density=0.6 in Random DCOP problems.

<table>
<thead>
<tr>
<th>nAgents</th>
<th>algorithm</th>
<th>communication time=0</th>
<th>communication time=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NCCCs</td>
<td>msgNumber</td>
<td>Runtime</td>
</tr>
<tr>
<td>6</td>
<td>ADOPT</td>
<td>13920</td>
<td>11469</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>111</td>
<td>1476</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>111</td>
<td>1476</td>
</tr>
<tr>
<td>8</td>
<td>ADOPT</td>
<td>422048</td>
<td>286313</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>225</td>
<td>5154</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>218</td>
<td>5075</td>
</tr>
<tr>
<td>10</td>
<td>ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>2345</td>
<td>82325</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>1194</td>
<td>6141</td>
</tr>
<tr>
<td>12</td>
<td>ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>7535</td>
<td>502112</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>5345</td>
<td>401716</td>
</tr>
<tr>
<td>14</td>
<td>ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>9687</td>
<td>412904</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>8602</td>
<td>409536</td>
</tr>
</tbody>
</table>
that Domains varies from 3 to 8 with density=0.4 and nAgents=10.

Firstly, as can be seen in Table 1 and Table 2, NCCCs in Layering-ADOPT is smaller than ADOPT and BnB-ADOPT in different cases of nAgents, especially when the communication time is 100. It can be also seen that Layering-ADOPT is superior to ADOPT and BnB-ADOPT over msgNumber and Runtime. Table 4 show the same trend for varying Domains. The improved performance of Layering-ADOPT comes from two facts: upper DFS-based agents can earlier prune sub-optimal branches; prompt feedbacks are provided by lower agents using BFS strategy.

From Table 3, the results show that Layering-ADOPT is better than ADOPT and BnB-ADOPT when density<0.7. However, there is an exceptional case when density=0.7. In this case, we find that the pseudo-tree degenerates into a chain since the constraint graph is very dense. Under this circumstance, the concurrency is almost lost for the lower BFS-based agents in the chain so that the reconstruction of BFS-based agents is aggravated. Accordingly, Layering-ADOPT performs worse than BnB-ADOPT. Therefore, Layering-ADOPT is not suitable for the very dense constraint graph.

5 CONCLUSION

In this paper, Layering-ADOPT is proposed to get a better tradeoff between ADOPT and BnB-ADOPT, where each agent selects its search strategy according to its position in the pseudo-tree and a layering boundary. The message processing and data structure in Layering-ADOPT are similar to those in ADOPT and BnB-ADOPT. Also, Layering-ADOPT realizes the joint between DFS and BFS strategy. It can be seen from the experimental results that Layering-ADOPT outperforms ADOPT and BnB-ADOPT in most cases. The improvement comes from the following points: firstly, the serious solution reconstructions are avoided in upper agents when upper agents use DFS strategy; secondly, the prompt feedback can be provided by lower agents when lower agents use BFS strategy.

In the future works, we hope to improve Layering-ADOPT by introducing a more suitable layering boundary and dynamically changeable search strategy instead of fixed one for each agent.

Table 3. The result of varying density with Domains=3 and nAgents=10 in Random DCOP problems.

<table>
<thead>
<tr>
<th>nAgents</th>
<th>algorithm</th>
<th>communication time=0</th>
<th>communication time=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NCCCs</td>
<td>msgNumber</td>
</tr>
<tr>
<td>0.4</td>
<td>ADOPT</td>
<td>708</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>558</td>
<td>18465</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.5</td>
<td>ADOPT</td>
<td>-</td>
<td>2474</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>1166</td>
<td>50527</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.6</td>
<td>ADOPT</td>
<td>-</td>
<td>4614</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>2192</td>
<td>43569</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.7</td>
<td>ADOPT</td>
<td>-</td>
<td>6573</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>7597</td>
<td>63758</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4. The result of varying Domains with density=0.4 and nAgents=10 in Random DCOP problems.

<table>
<thead>
<tr>
<th>nAgents</th>
<th>algorithm</th>
<th>communication time=0</th>
<th>communication time=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NCCCs</td>
<td>msgNumber</td>
</tr>
<tr>
<td>4</td>
<td>ADOPT</td>
<td>821</td>
<td>21156</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>789</td>
<td>20277</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>ADOPT</td>
<td>1437</td>
<td>38709</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>1217</td>
<td>35169</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>ADOPT</td>
<td>1862</td>
<td>49320</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>1580</td>
<td>46250</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>ADOPT</td>
<td>2972</td>
<td>79419</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>2585</td>
<td>74025</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>ADOPT</td>
<td>7092</td>
<td>184899</td>
</tr>
<tr>
<td></td>
<td>BnB-ADOPT</td>
<td>6592</td>
<td>179882</td>
</tr>
<tr>
<td></td>
<td>Layering-ADOPT</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

This work is supported by the Chongqing University Postgraduates’ Innovation Project (Project No. CYS14018).

REFERENCES


